IPv6 for the masses... with 6bed4

OpenFortress*
digital signatures

introduction

* 6bed4 is Yet Another Tunnel?!?
  → Arose from practical needs
  → SIP over IPv6 *anywhere*
  → No existing tunnel seemed suitable
  → New requirements: zero-config, peer-to-peer
  → 6bed4 builds on experience with previous tunnels

* Turned out to be generally useful
  → Zero-config means *just install & run*
  → Peer-to-peer means *scalable tunneling*

introduction

* New approach to peer-to-peer direct traffic
  - → No classification of NAT
  - → Simply *try* to pass traffic directly
  - → Rely on a fallback service for failing peers

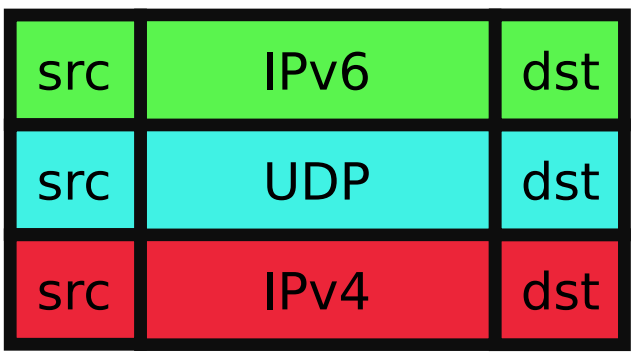
WikiMedia Commons

requirements and protocol choices

work behind any (nat) router sequence

* **Requirement:** Do not assume co-operation from a router
  → Facilitating internal 6bed4 for hosts and (embedded) devices

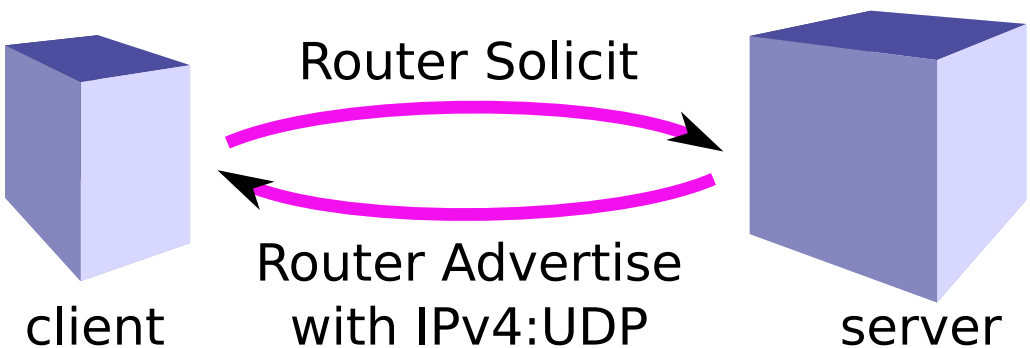
* **Choice:** Run IPv6 over UDP/IPv4
  → Only assumption made is permitted outgoing UDP



open and simple

* **Issue:** How to obtain your local IPv6 address?
  → Options: Own protocol, STUN, DHCPv6, SLAAC, . . .

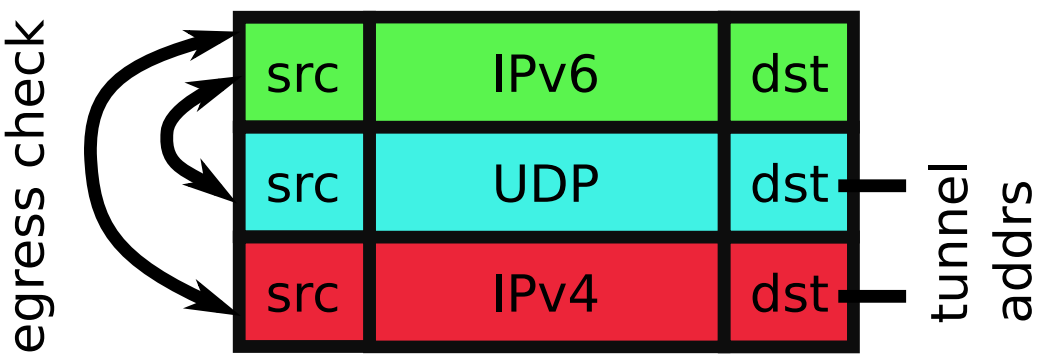
* **Choice:** Use Router Discovery
  → Small adaption: Supply external IPv4:UDP to client



abuse tracking

* **Requirement:** Be able to track down network abuse
  → Options: ISP-locality, accounts, embedded IPv4, . . .

* **Choice:** Embed IPv4 address into IPv6
  → 6bed4 will also embed the UDP port
  → Use source IPv4:UDP for IPv6 ‘egress’ filtering



zero configuration

* **Desire:** No configuration needed by end users
  → This means that IPv6 is never a hurdle to them
  → As a result, no obstructions to building it into devices/distros

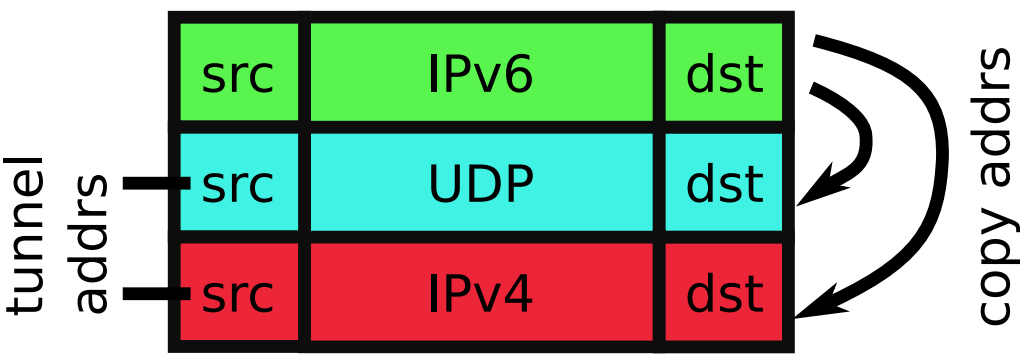
* **Option:** Configure a well-known service address

* **Option:** Do not depend on user accounts

stateless tunnel service

* **Desire:** Tunnel service should be stateless
  → Sensible for routing: simpler traffic diversion

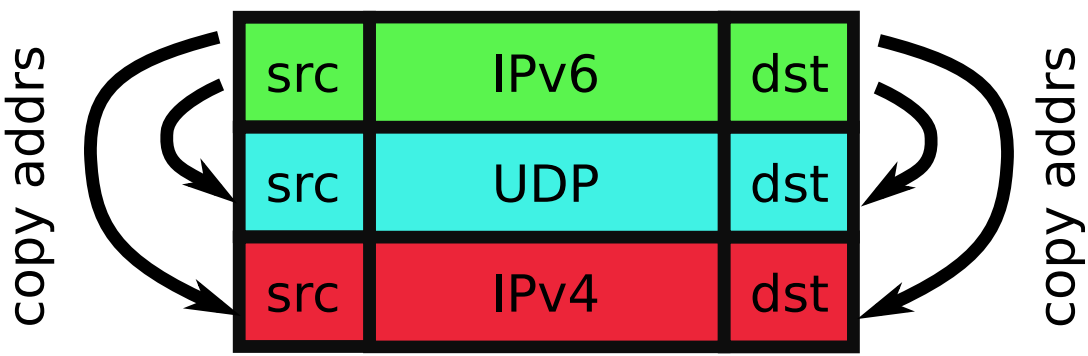
* **Choice:** Embed IPv4:UDP in client-side IPv6 address
  → Use IPv4:UDP to determine how to forward IPv6 traffic



scalability: optional bypass for return traffic

* **Desire:** Servers can install 6bed4 as a return traffic path
  → This means they pack an IPv6 answer into UDP/IPv4

* **Option:** Setup a well-known IPv6 prefix for 6bed4 traffic
  → A server may setup a 6bed4 interface to handle that prefix
  → It *might* be able to reply directly to the sender...



scalability: direct contact between 6bed4 peers

* **Requirement:** Peers get in direct contact
  → Any public service should merely be a fallback option
  → Symmetric data transfer (client ≡ server) is desired

* **Choice:** Contact peer directly on their IPv4:UDP
  → This information is available in the IPv6 address
  → The well-known IPv6 prefix for 6bed4 makes it recognisable

* **Choice:** Knock on the peer's door with Neighbor Discovery
  → Bidirectional ICMPv6 works ⇒ then bidirectional IPv6 works
  → Bidirectional traffic causes both sides to attempt this
  → Symmetric NAT is the only expected part to fail
  → Carrier Grade NAT is not expected to be symmetric (for UDP)

relation to the ipv6 stack

* **Requirement:** Keep 6bed4 simple, in spite of changeable routes
* **Requirement:** Make no changes to the IPv6 stack

* An IPv6 stack would see 6bed4 as its link-local layer
  → The IPv4:UDP are the link-local addresses on that network
  → These can be stored in the Neighbor Cache
  → If NAT traversal fails, the tunnel server's IPv4:UDP is used

* Enjoy the facilities of the Neighbor Cache
  → Neighbor Discovery triggers attempts to route peer-to-peer
  → Repeated Neighbor Discovery keeps trying NAT-traversal
  → Trigger Neighbor Discovery if incoming traffic follows a shorter path
  → Mind security (filter on sender) on incoming Neighor Solicitations

experimenting with anycast

experimenting with anycast

* **Desire:** Use anycast addresses for IPv4 and 6bed4 prefix

* IPv4 is `145.136.0.1`

* IPv6 prefix is `2001:67c:127c::/48`

experimenting with anycast

* **Problem?** Return routes are uncontrollable

* Geoff Huston found these OK when testing 6to4

* Any servers could setup a local 6bed4 interface

* En-route translation *might* use local IPv6 ranges

* We *could* use `145.136.0.2` to run under a local IPv6 prefix
  $\rightarrow$ But then: Route change $\Rightarrow$ connection breakdown

experimenting with anycast

* **Problem?** It is hard to monitor anycast services

* Nothing stops us from adding a local IPv4 and IPv6 prefix

* Software on the same host could serve multiple address pairs

* Just monitor a local IPv4 address and IPv6 prefix

experimenting with anycast

* **Problem?** The cost of anycast services are uncontrollable

* Anycast routes are published over BGP4

* Just control who may route their 6bed4 traffic to you

* Stateless translation could be one-sided

* 6bed4 could easily be kept ISP-local

placing the work in context

not all requirements are fulfilled yet

Goal	6in4	6to4	S'wire	TSP	Teredo	AYIYA	TURN	6bed4
Standard	√	√	√	±	±	×	√	√
Simple	√	√	×	√	×	√	√	√
Any router	×	×	√	√	×	√	√	√
No config	×	√	×	√	√	×	×	√
Tracking	×	√	√	?	?	√	×	√
Peer2peer	×	√	×	×	×	×	×	√
Stateless	√	√	×	×	×	×	×	√
Anycast	×	√	×	×	√	×	×	√
Symmetry	×	√	×	×	×	×	×	√

work in progress

* Software on `http://devel.0cpm.org/6bed4/`

* Internet Draft awaits your comments
  → Lists at `http://sf.net/p/tun6bed4/mailman/`

* First public service node is kindly provided by SURFnet

* Getting from v00 to v01 is funded by OpenFortress and NLnet

* We will be testing anycast performance and issues next year
  → Parties involved in routing are invited to join in

* SURFnet prepares a thorough comparison of tunneling protocols
  → Will take 6bed4 into account

conclusions

conclusions

* Tunneling SIP/RTP introduces new requirements
* These requirements are generally useful
* Existing tunnels leave requirements unfulfilled
* Currently, 6bed4 resolves the identified requirements

* As far as we are concerned. . . *6bed4 is to serve the masses*

info@openfortress.nl

http://openfortress.nl

OpenFortress*

digital signatures